Diffusion Models: Ron's Take
Gian Favero — October 2nd, 2023

McGill

1 Denoising Diffusion Probabilistic Models

A denoising diffusion model (DDPM) contains a neural network that learns to gradually denoise data starting from pure noise into a final image. The set-up consists of two processes:

1. A fixed and pre-defined forward diffusion process q of our choosing, that gradually adds Gaussian noise to an image, until you end up with pure noise
2. A learned reverse denoising diffusion process p_θ where a neural network is trained to gradually denoise an image starting from pure noise, until you end up with an actual image

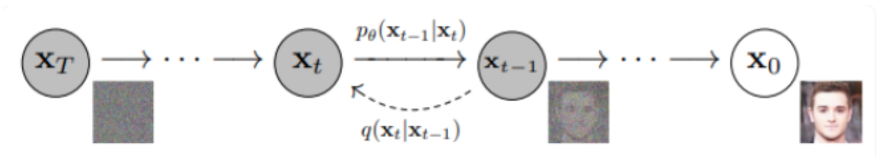

$\mathbf{x}_T$
$\mathbf{x}_t$
$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$
$\mathbf{x}_{t-1}$
$q(\mathbf{x}_t|\mathbf{x}_{t-1})$
$\mathbf{x}_0$

Figure 1: Graphical representation of a diffusion model

Both forward and reverse processes are indexed by t over a number of time-steps T (in this paper, $T = 1000$). At $t = 0$, a real image $\mathbf{x}_0$ is sampled from a data distribution $q(\mathbf{x}_0)$. At each time-step t, the forward process adds Gaussian noise to the image such that at a sufficiently large T and a well-behaved noise schedule, a Gaussian distribution centered at 0 with variance of 1 is obtained at $t = T$.

1.1 Mathematical Formulation

1.1.1 Forward Process

Let $q(\mathbf{x}_0)$ be a real data distribution from which we sample a real image, $\mathbf{x}_0 \sim q(\mathbf{x}_0)$. A forward process is defined as $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ which adds Gaussian noise at each time-step according to a pre-defined noise schedule $0 \leq \beta_1 \leq \beta_2 \leq ... \leq \beta_T \leq 1$.

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

Basically, each new image at time-step t is drawn from a conditional Gaussian distribution with $\mu_t = \sqrt{1-\beta_t}\mathbf{x}_{t-1}$ and $\sigma_t^2 = \beta_t$. Note that the mean is slightly scaled down at each time-step to preserve the norm of the image when noise is added.

In a more succinct form, the forward process can be written as:

$$\mathbf{x}_t = \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\epsilon, \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

β_t is not constant at each time-step and follows a "variance schedule" which can be linear, quadratic, cosine, etc. In any case, starting from $\mathbf{x}_0$, we end up with $\mathbf{x}_T$ which is pure noise if the schedule is set up appropriately. Holding β_t constant allows for the posterior q to have no learnable parameters.

1.1.2 Reverse Process

If the conditional distribution $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ was known, we could sample some noise $\mathbf{x}_T$ and then gradually denoise it until we end up with a real image $\mathbf{x}_0$. However, this conditional distribution is not known and is instead learned by a neural network $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$, where θ are the parameters of the neural network.

Assuming that the reverse process is also Gaussian, we can write the reverse process as:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_t^2\mathbf{I})$$

Where the mean $\mu_\theta(\mathbf{x}_t, t)$ is conditioned on the noise level t and the variance is held constant at $\sigma_t^2 = \beta_t$. A neural network is trained to learn the mean $\mu_\theta(\mathbf{x}_t, t)$.

1.1.3 Optimizing the Variational Lower Bound

Training is performed by optimizing the variational bound on the negative log-likelihood:

$$\begin{aligned}\mathbb{E}[-\log p_\theta(\mathbf{x}_0)] &\leq \mathbb{E}_q\left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] \\ &= \mathbb{E}_q\left[-\log p(\mathbf{x}_T) - \sum_{t\geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})}\right]\end{aligned}$$

This can be further deconstructed into:

$$\mathbb{E}_q\Big[\underbrace{D_{KL}(q(\mathbf{x}_T|\mathbf{x}_0)||p(\mathbf{x}_T))}_{L_T} + \sum_{t>1}\underbrace{D_{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} - \underbrace{\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0}\Big]$$

L_T is a constant and can be ignored because the forward process is predefined with the noise variance schedule β_t.

L_0 is the standard reconstruction term found in variational auto-encoders. An independent discrete decoder derived from the Gaussian $\mathcal{N}(\mathbf{x}_0; \mu_\theta(\mathbf{x}_1, 1), \sigma_1^2\mathbf{I})$ is used to reconstruct the image $\mathbf{x}_0$ from $\mathbf{x}_1$.

L_{t-1} is the loss term of particular interest. It is the distance between the desired denoising steps $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ and the approximated denoising steps $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$. This is the term that is minimized during training.

1.2 Training

1.2.1 Objective Function

To derive an objective function we can look at the combination of q and p_θ as a variational auto-encoder. The variational lower bound (or ELBO) is used to minimize the KL-divergence between the learned posterior $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$, and the intractable posterior, $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$.

Since the forward process is a repeated summation of Gaussian noise, a sample at time-step t can be written as a conditional distribution of $\mathbf{x}_0$, since the sum of Gaussians is also Gaussian. This is a simple property that allows us to add any scaled Gaussian noise to $\mathbf{x}_0$ and end up with $\mathbf{x}_t$ for any t.

$$\mathbf{x}_t = \sqrt{\overline{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\overline{\alpha}_t}\epsilon, \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

where $\alpha_t = 1 - \beta_t$ and $\overline{\alpha}_t = \prod_{s=1}^{t} \alpha_s$. The intractable posterior can in turn be made tractable through conditioning it on $\mathbf{x}_0$.

$$\begin{aligned} q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) &= \mathcal{N}(\mathbf{x}_{t-1}; \widetilde{\mu}_t(\mathbf{x}_t), \widetilde{\beta}_t\mathbf{I}) \\ \widetilde{\mu}_t(\mathbf{x}_t) &= \frac{1}{\sqrt{\alpha_t}}(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\overline{\alpha}_t}}\epsilon) \end{aligned}$$

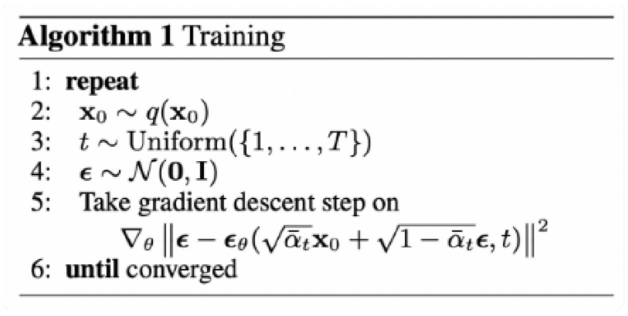
Ideally, an estimator,

$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\overline{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t))$$

can be learned to approximate $\widetilde{\mu}_t(\mathbf{x}_t)$. A neural network can be trained to estimate the noise using the objective function:

$$\begin{aligned} L_t &= \mathbb{E}_{\mathbf{x}_0,t,\epsilon}[\| \widetilde{\mu}_t(\mathbf{x}_t) - \mu_\theta(\mathbf{x}_t, t) \|_2^2] \\ &= \mathbb{E}_{\mathbf{x}_0,t,\epsilon}\left[\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\overline{\alpha}_t)} \| \epsilon - \epsilon_\theta(\mathbf{x}_t, t) \|_2^2\right] \\ &\approx \mathbb{E}_{\mathbf{x}_0,t,\epsilon}\left[\| \epsilon - \epsilon_\theta(\mathbf{x}_t, t) \|_2^2\right] \end{aligned}$$

The proof for this can be found here.



```
Algorithm 1 Training
1: repeat
2:   $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3:   $t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
4:   $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5:   Take gradient descent step on
         $\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$
6: until converged
```

Figure 2: DDPM Training Process

In simple terms,

1. A real image $\mathbf{x}_0$ is sampled from a data distribution $q(\mathbf{x}_0)$
2. A noise level t is sampled from a uniform distribution $t \sim \mathcal{U}(0, T)$
3. A noise ϵ is sampled from a Gaussian distribution $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
4. The neural network is trained to predict this noise based on the corrupted image $\mathbf{x}_t$ (noise applied to $\mathbf{x}_0$ based on the known schedule β_t)

1.2.2 Network Architecture

The neural network needs to take in a noised image $\mathbf{x}_t$ and output a noise $\epsilon_\theta(\mathbf{x}_t, t)$ that was added to the image. As both the input and output tensors are of the same size and resolution, a U-Net architecture is used.

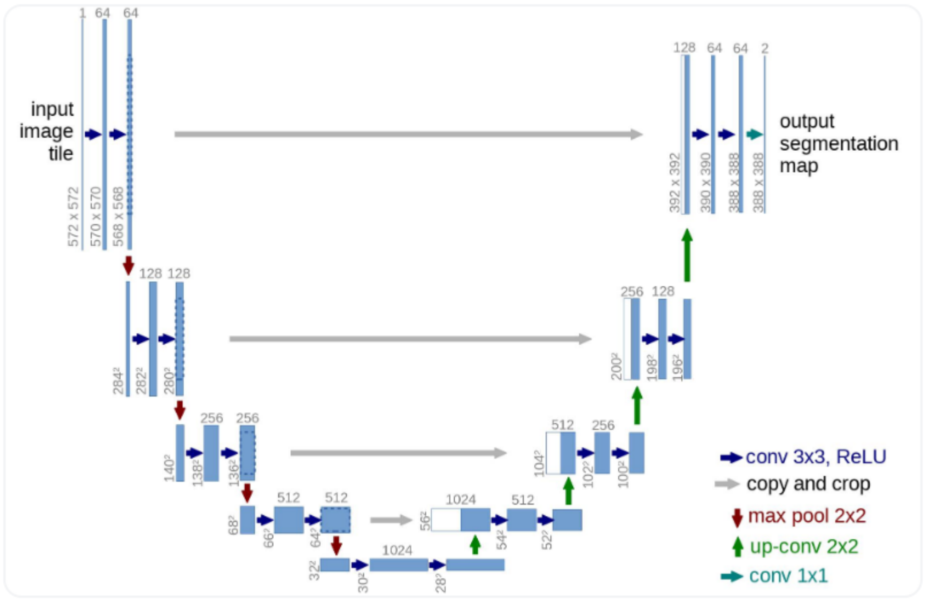
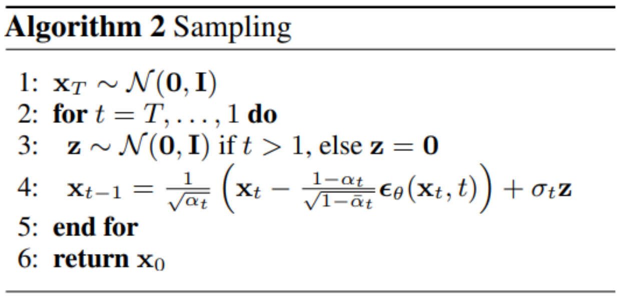


Figure 3: U-Net Architecture

Recalling that $\mathbf{x}_t = \sqrt{\overline{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\overline{\alpha}_t}\epsilon$, the network is aware of the noise schedule β_t and is fed sinusoidal positional embeddings of the noise level t in the skip connections. Based on the noise schedule, the noise level, and the noised image, the network can eventually learn to predict the noise that was added at t.

1.3 Sampling

Once the network is trained, we can sample from the model by starting with pure noise $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and gradually denoise it using $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ until we end up with a real image $\mathbf{x}_0$.



Algorithm 2 Sampling

```
1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: for $t = T, \ldots, 1$ do
3:   $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4:   $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t \mathbf{z}$
5: end for
6: return $\mathbf{x}_0$
```

Figure 4: DDPM Sampling Process

It can be shown that at each step:

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)) + \sigma_t \mathbf{z}, \quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

And therefore starting from $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, we can iteratively generate $\mathbf{x}_{T-1}$, then $\mathbf{x}_{T-2}$, and so on until we end up with $\mathbf{x}_0$.

2 Latent Diffusion Models

Pixel-based diffusion models are prone to spending excessive amounts of capacity (and thus compute resources) on modeling imperceptible details of the data. A departure to the latent space can help alleviate this problem.

2.1 Method

To lower the computational demands of training diffusion models towards high-resolution image synthesis, it is observed that although diffusion models allow perceptually irrelevant details to be ignored by undersampling the corresponding loss terms, they still require costly function evaluations in the pixel space, which causes huge demands in computation time and energy resources.

To circumvent this problem, a multi-part approach is proposed:

1. A universal autoencoder is trained to compress an image into a latent space, and then reconstruct it back to its original form
2. A diffusion model is trained in the latent space of the autoencoder

2.2 Departure to Latent Space

In finding a latent representation of a high-resolution image, the aim lies in finding a perceptually equivalent but computationally more suitable space for training a diffusion model. An autoencoder is a perfect candidate for this task, as it can be trained to compress an image into a latent space, and then reconstruct it back to its original form. In this manner, a universal autoencoder can be trained once and then used for many subsequent diffusion model training tasks.

Given an image $x \in \mathbb{R}^{H \times W \times 3}$, the encoder compresses it into a latent space $z = \mathcal{E}(x)$, where $z \in \mathbb{R}^{h \times w \times c}$and a decoder reconstructs the image from the latent, $\tilde{x} = \mathcal{D}(z)$. The encoder downsamples the original image by a factor $f = H/h = W/w$, with different factors ($f = 2^m, m \in \mathbb{N}$) producing varying results.

The autoencoder is trained to minimize the reconstruction loss between the original image and the reconstructed image. Two regularizations can be enforced: KL-reg and VQ-reg. KL-reg imposes a KL-penalty towards a standard normal on the learned latent, while VQ-reg uses a vector quantization layer within the decoder.

2.3 Generative Modelling in the Latent Space

With the encoder there is now access to an efficient, low-dimensional latent space in which high-frequency, imperceptible details are abstracted away. Compared to the high-dimensional pixel space, this space is more suitable for likelihood-based generative models, as they can now (i) focus on the important, semantic bits of the data and (ii) train in a lower dimensional, computationally much more efficient space.

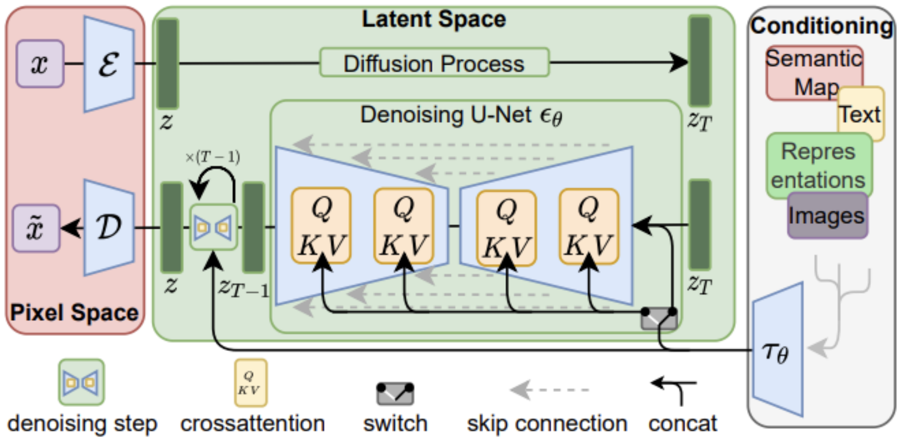


Figure 5: Latent Diffusion Model Architecture

The underlying UNet architecture is trained on the latent space with a loss function that is similar to the one used in the original DDPM paper:

$$L_{LDM} = \mathbb{E}_{\mathcal{E}(x),\epsilon\sim\mathcal{N}(0,1),t}[\| \ \epsilon - \epsilon_\theta(z_t, t) \ \|_2^2]$$

Since the forward process is fixed, z_t can be efficiently obtained from $\mathcal{E}$ during training, and samples from $p(z)$ can be decoded to the image space with a single pass through $\mathcal{D}$.

2.4 Conditioning Mechanisms

The latent diffusion model can produce conditional distributions of the form $p(z|y)$ where y is a conditioning variable like text, semantic maps, or other images (e.g., super-resolution). This is achieved by conditioning the denoising autoencoder $\epsilon_\theta(z_t, t, y)$.

To implement this, the UNet backbone is augmented with a cross-attention mechanism that allows the model to learn various input modalities. A domain specific encoder, τ_θ is introduced to project y to an embedding space $\tau_\theta(y) \in \mathbb{R}^{M\times d_\tau}$, which is then mapped to the intermediate layers of the UNet via cross-attention layers. The LDM paper can be referenced for more details.

Based on image-conditioning pairs, the conditional LDM can be learned via:

$$L_{LDM} = \mathbb{E}_{\mathcal{E}(x),\epsilon\sim\mathcal{N}(0,1),t}[\| \ \epsilon - \epsilon_\theta(z_t, t, \tau_\theta(y)) \ \|_2^2]$$

2.5 Experimental Findings

2.5.1 Image Compression

Small downsampling factors ($f \in \{1, 2\}$) result in slow training progress, while large values ($f > 16$) result in stagnating fidelity after comparably few training steps. The best results are achieved using a downsampling factor in the range of $f \in \{4, 8, 16\}$.

An analysis of reasoning here can be attributed to where the perceptual compression takes place. With low downsampling factors, the diffusion model is responsible for most of the compression, which is a slow process. With high downsampling factors, lots of information is lost in the encoding process, resulting in a loss of fidelity.

2.5.2 Super-Resolution

LDMs can be efficiently trained for super-resolution by diretly conditioning on low-resolution images via concatenation. Images are degraded using a bicubic downsampling kernel (4x downsampling) and then fed into the LDM as a low-resolution conditioning, y along with its identity, τ_θ.

This process allows the upscaling of 256^2 images to 1024^2 images with a single forward pass through the LDM. The results are comparable to the state-of-the-art super-resolution methods (SR3, image regression), but with a much more efficient training process.